ΣΧΕΔΙΑΣΜΌΣ ΣΥΣΤΗΜΆΤΩΝ **VLSI**

$4^{η}$ Εργαστηριακή Άσκηση

Γιώργος Ντάκος 1059569

26 Μαΐου 2021

Περίληψη

Στο παρακάτω κείμενο παραδίδεται η αναφορά της $4^{ης}$ εργαστηριακής άσκησης του μαθήματος Σχεδιασμός Συστημάτων VLSI . Όλοι οι κώδικες έχουν συγγραφεί σε γλώσσα **VHDL** μέσω του **Notepad++** και έχουν μεταγλωτειστεί και επιβεβαιωθεί οι λειτουργίες τους μέσω του εργαλείου **ModelSim**.Τέλος η σύνθεση όλων των κυκλωμάτων πραγματοποιήθηκαν από το εργαλείο **VIVADO** της **Xilnix**.

Περιεχόμενα

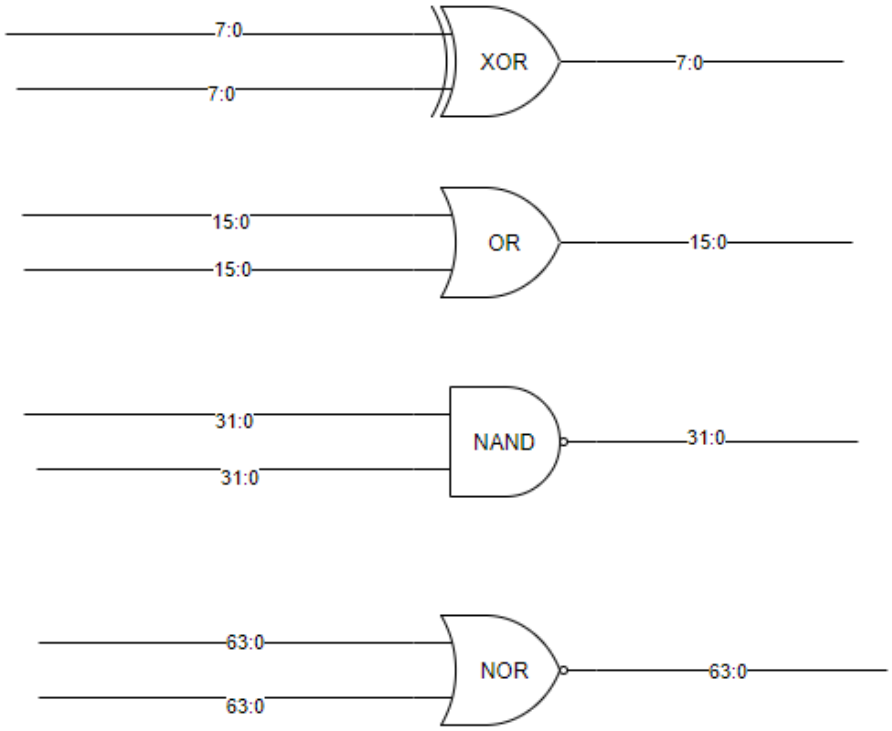
Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Ενότητα Α

Α.1,Α.2

Στην συγκεκριμένη ενότητα θα φτιάξουμε 4 λογικές πύλες δύο εισόδων και μιας εξόδου, (με διαφορετικά πλάτοι για την κάθε πύλη) οι οποίες φαίνονται στο παρακάτω σχηματικό και στον πίνακα 1 βρίσκονται οι πίνακες αληθείας τους

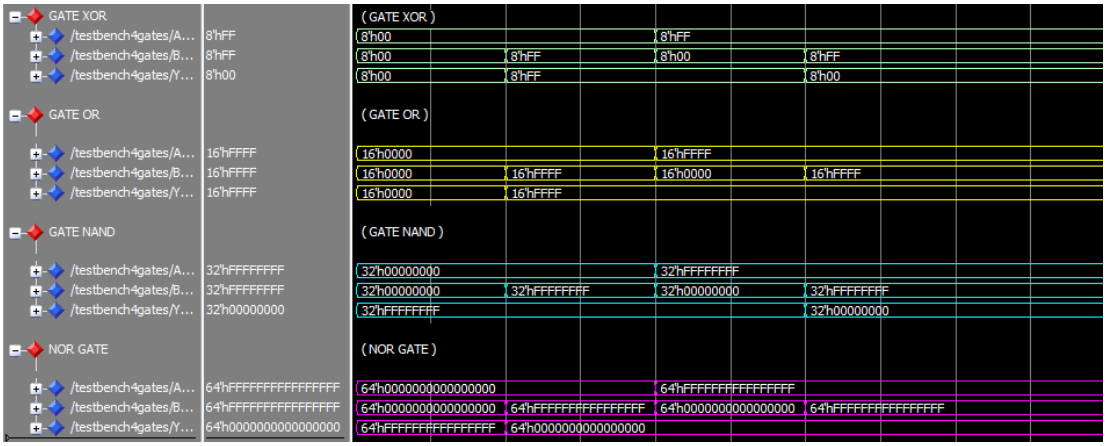


Σχήμα 1: Σχηματικό των λογικών πυλών

Στην συνέχεια ακολουθούν τα αποτελέσματα από το simulation μέσω ενός testbench για τις 4 λογικές πυλές και τα αποτελέσματα σύνθεσης για την κάθε μια.

GATE : XOR	IN_A	IN_B	OUT
	0	0	0
	0	1	1
	1	0	1
	1	1	0
GATE : OR	**IN_A**	**IN_B**	**OUT**
	0	0	0
	0	1	1
	1	0	1
	1	1	1
GATE : NAND	**IN_A**	**IN_B**	**OUT**
	0	0	1
	0	1	1
	1	0	1
	1	1	0
GATE : NOR	**IN_A**	**IN_B**	**OUT**
	0	0	1
	0	1	0
	1	0	0
	1	1	0

Table 1: Πίνακες αληθείας



Σχήμα 2: Αποτελέσματα simulation

Circuit	*LUTs*	*IO*	*FlipFlops*	*BRAM*	*Critical Path (ns)*	*Clock (MHz)*	*Fmax(MHz)*
xor8	8(0.04%)	24(11.43%)	0	0	4.321	0	231.427
or16	16(0.08%)	48(22.86%)	0	0	4.321	0	231.427
nand32	32(0.15%)	96(45.71%)	0	0	4.321	0	231.427
nor64	64(0.31%)	192(91.43%)	0	0	4.321	0	231.427

Table 2: Αποτελέσματα Σύνθεσης των λογικων πυλών XOR, OR, NAND, NOR

Παρατηρείται ότι ο αριθμός των εξωτερικών σημάτων I/O αυξάνεται γραμμικά με το πλάτος κάθε πύλης. Κάθε πύλη διαθέτει δύο εισόδους και μία έξοδο εύρους N bit, επομένως απαιτούνται συνολικά $2N + N = 3N$ ακροδέκτες I/O. Έτσι προκύπτουν: $N = 8 \rightarrow 24$ I/O, $N = 16 \rightarrow 48$ I/O, $N = 32 \rightarrow 96$ I/O, $N = 64 \rightarrow 192$ I/O, τιμές που συμφωνούν πλήρως με τα αποτελέσματα της σύνθεσης.

Η στήλη *Clock (MHz)* εμφανίζεται ως μηδενική, καθώς δεν έχει οριστεί χρονικός περιορισμός (XDC constraint) στο κύκλωμα. Οι πύλες είναι πλήρως συνδυαστικές και η μέγιστη θεωρητική συχνότητα λειτουργίας προκύπτει από το κρίσιμο μονοπάτι:

$$F_{\max} \approx \frac{1}{T_{critical}}.$$

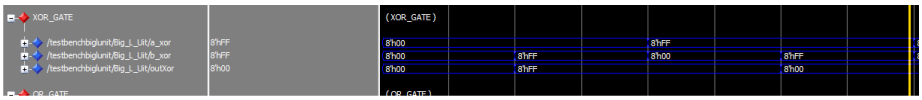
Ενότητα B

B.1,B.2

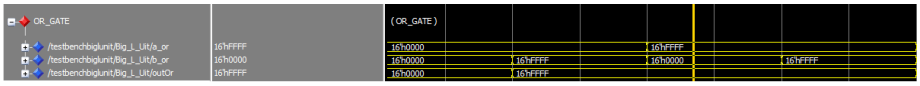
Η μονάδα `bigLunit` διαθέτει δύο εισόδους των 120-bit, μία έξοδο των 120-bit και ένα σήμα επιλογής 2-bit, συνεπώς απαιτεί συνολικά

$$120 + 120 + 120 + 2 = 362$$

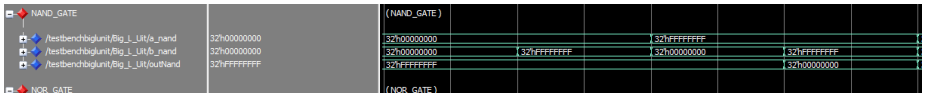
εξωτερικά I/O. Στο αρχικά επιλεγμένο FPGA (`xc7a35tcsg324-3`) ο διαθέσιμος αριθμός User I/O δεν επαρκεί, με αποτέλεσμα η υλοποίηση να εμφανίζει υπέρβαση I/O (*IO utilization* $> 100\%$). Για τον λόγο αυτό επιλέχθηκε συσκευή με πακέτο που διαθέτει τουλάχιστον 362 I/O, ώστε να καταστεί δυνατή η πλήρης υλοποίηση και η εξαγωγή έγκυρων αποτελεσμάτων σύνθεσης.



Σχήμα 3: Επιλογή πύλης XOR και τα αποτέλεσματα του simulation



Σχήμα 4: Επιλογή πύλης 0R και τα αποτελέσματα του simulation

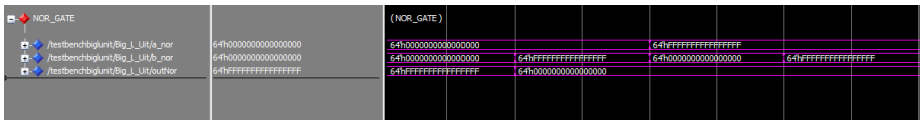


Σχήμα 5: Επιλογή πύλης NAND και τα αποτελέσματα του simulation

Με τη νέα επιλογή συσκευής, η `bigLunit` χρησιμοποιεί 362 ακροδέκτες I/O, που αντιστοιχούν σε ποσοστό 90.50% των διαθέσιμων ακροδεκτών του πακέτου, επομένως η σχεδίαση χωράει οριακά ως προς τα εξωτερικά σήματα.

Οι τιμές του κρίσιμου μονοπατιού της `bigLunit` είναι συγκρίσιμες με εκείνες των επιμέρους πυλών, καθώς πρόκειται για συνδυαστικό κύκλωμα. Η μέγιστη συχνότητα λειτουργίας προκύπτει από τη σχέση $F_{\max} \approx 1/T_{critical}$.

Θεωρητικά, η ενσωμάτωση της πύλης στη `bigLunit` μέσω πολυπλέκτη εισάγει επιπλέον λογική βαθμίδα και θα αναμενόταν ίση ή μεγαλύτερη καθυστέρηση. Ωστόσο, στο FPGA η συνολική καθυστέρηση κυριαρχείται από το φυσικό routing. Με τη χρήση πακέτου με περισσότερα I/O και την ομαδοποίηση της πύλης και του πολυπλέκτη σε γειτονικούς πόρους, το Vivado μειώνει τις φυσικές διαδρομές από τα pins προς τη λογική. Έτσι, το routing delay μειώνεται περισσότερο από όσο αυξάνεται η καθυστέρηση λόγω του MUX, με αποτέλεσμα η `bigLunit` να εμφανίζει ελαφρώς μικρότερο κρίσιμο μονοπάτι.



Σχήμα 6: Επιλογή πύλης NOR και τα αποτελέσματα του simulation

Circuit	*LUTs*	*IO*	*FlipFlops*	*BRAM*	*Critical Path (ns)*	*Clock (MHz)*	*Fmax(MHz)*
bigLunit	120(0.09%)	362(90.50%)	0	0	4.275	0	233.918

Table 3: Αποτελέσματα Σύνθεσης του bigLunit

Κώδικες σε VHDL

Παρακάτω παρατίθονται οι κώδικες της εργασίας(στο τέλος ακολουθούν τα σχηματικά σύμφωνα με την σύνθεση του κάθε κώδικα):

---

xor8

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

ENTITY xor8 IS
GENERIC(N: NATURAL :=8);
PORT(

 A :IN std_logic_vector(N-1 DOWNTO 0);

 B :IN std_logic_vector(N-1 DOWNTO 0);

 Y :OUT std_logic_vector(N-1 DOWNTO 0));
 END xor8;

 ARCHITECTURE structural OF xor8 IS
 BEGIN

 XORLoop: For i IN 0 TO N-1 Generate

 Y(i) <= A(i) XOR B(i);

 END Generate XORLoop;
 END structural;
```

---

or16

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

ENTITY or16 IS
GENERIC(N: NATURAL :=16);
PORT(

 A :IN std_logic_vector(N-1 DOWNTO 0);

 B :IN std_logic_vector(N-1 DOWNTO 0);

 Y :OUT std_logic_vector(N-1 DOWNTO 0));
 END or16;

 ARCHITECTURE structural OF or16 IS
 BEGIN

 ORLoop: FOR i IN 0 TO N-1 GENERATE

 Y(i) <= A(i) OR B(i);

 END GENERATE ORLoop;
 END structural;
```

nand32

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

ENTITY nand32 IS
GENERIC(N: NATURAL :=32);
PORT(

 A :IN std_logic_vector(N-1 DOWNTO 0);

 B :IN std_logic_vector(N-1 DOWNTO 0);

 Y :OUT std_logic_vector(N-1 DOWNTO 0));
 END nand32;

 ARCHITECTURE structural OF nand32 IS
 BEGIN
```

```
 NANDLoop: For i IN 0 TO N-1 Generate

 Y(i) <= A(i) NAND B(i);

 END Generate NANDLoop;
 END structural;
```

---

nor64

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

ENTITY nor64 IS
GENERIC(N: NATURAL :=64);
PORT(

 A :IN std_logic_vector(N-1 DOWNTO 0);

 B :IN std_logic_vector(N-1 DOWNTO 0);

 Y :OUT std_logic_vector(N-1 DOWNTO 0));
 END nor64;

 ARCHITECTURE structural OF nor64 IS
 BEGIN

 NORLoop: FOR i IN 0 TO N-1 GENERATE

 Y(i) <= A(i) NOR B(i);

 END GENERATE NORLoop;
 END structural;
```

---

testbench4GATES

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY testbench4GATES IS
END testbench4GATES;

ARCHITECTURE testbench OF testbench4GATES IS
    COMPONENT xor8
    GENERIC(
        N: NATURAL :=8);
```

```
PORT(
 A : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
 B : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
 Y : OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0)
);
END COMPONENT;

COMPONENT or16
GENERIC(
N: NATURAL :=16);
PORT(
 A : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
 B : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
 Y : OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0)
);
END COMPONENT;

COMPONENT nand32
GENERIC(
N: NATURAL :=32);
PORT(
 A : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
 B : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
 Y : OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0)
);
END COMPONENT;

COMPONENT nor64
GENERIC(
N: NATURAL :=64);
PORT(
 A : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
 B : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
 Y : OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0)
);
END COMPONENT;

CONSTANT K : NATURAL := 8 ;
CONSTANT L : NATURAL := 16 ;
CONSTANT M : NATURAL := 32 ;
CONSTANT N : NATURAL := 64 ;

signal ATxor : STD_LOGIC_VECTOR(K-1 DOWNTO 0);
signal BTxor : STD_LOGIC_VECTOR(K-1 DOWNTO 0);
signal ATor : STD_LOGIC_VECTOR(L-1 DOWNTO 0);
signal BTor : STD_LOGIC_VECTOR(L-1 DOWNTO 0);
```

```
        signal ATnand : STD_LOGIC_VECTOR(M-1 DOWNTO 0);
        signal BTnand : STD_LOGIC_VECTOR(M-1 DOWNTO 0);
        signal ATnor : STD_LOGIC_VECTOR(N-1 DOWNTO 0);
        signal BTnor : STD_LOGIC_VECTOR(N-1 DOWNTO 0);

        signal YTxor : STD_LOGIC_VECTOR(K-1 DOWNTO 0);
        signal YTor : STD_LOGIC_VECTOR(L-1 DOWNTO 0);
        signal YTnand : STD_LOGIC_VECTOR(M-1 DOWNTO 0);
        signal YTnor : STD_LOGIC_VECTOR(N-1 DOWNTO 0);

BEGIN


        NOR_GATE: nor64 GENERIC MAP(N) PORT MAP(ATnor,BTnor,YTnor);
        NAND_GATE: nand32 GENERIC MAP(M) PORT MAP(ATnand,BTnand,YTnand);
        OR_GATE: or16 GENERIC MAP(L) PORT MAP (ATor,BTor,YTor);
        XOR_GATE: xor8 GENERIC MAP(K) PORT MAP(ATxor,BTxor,YTxor);

        stim_proc: process
        BEGIN

        ATxor<=X"00";
        BTxor<=X"00";

        ATor<=X"0000";
        BTor<=X"0000";

        ATnand<=X"00000000";
        BTnand<=X"00000000";

        ATnor<=X"0000000000000000";
        BTnor<=X"0000000000000000";

                WAIT FOR 20 ns;

        ATxor<=X"00";
        BTxor<=X"FF";

        ATor<=X"0000";
        BTor<=X"FFFF";

        ATnand<=X"00000000";
        BTnand<=X"FFFFFFFF";

        ATnor<=X"0000000000000000";
        BTnor<=X"FFFFFFFFFFFFFFFF";
```

```
                WAIT FOR 20 ns;

        ATxor<=X"FF";
        BTxor<=X"00";

        ATor<=X"FFFF";
        BTor<=X"0000";

        ATnand<=X"FFFFFFFF";
        BTnand<=X"00000000";

        ATnor<=X"FFFFFFFFFFFFFFFF";
        BTnor<=X"0000000000000000";

                WAIT FOR 20 NS;

        ATxor<=X"FF";
        BTxor<=X"FF";

        ATor<=X"FFFF";
        BTor<=X"FFFF";

        ATnand<=X"FFFFFFFF";
        BTnand<=X"FFFFFFFF";

        ATnor<=X"FFFFFFFFFFFFFFFF";
        BTnor<=X"FFFFFFFFFFFFFFFF";

                WAIT;

        END PROCESS;

END;
```

---

mux

```
--Dhlwsh bibliothhkhs kai twn paketwn poy tha xrhsimopoihsoyme
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

ENTITY mux IS
GENERIC(N: NATURAL :=8);
```

```
PORT (
-- Shma eisodoy poy lambanei mia akoloythia timwn.
        a:IN std_logic_vector(N-1 downto 0);
-- Shma eisodoy poy lambanei mia akoloythia timwn.
        sel:IN std_logic_vector(1 downto 0);
-- Shma eisodoy poy lambanei mia akoloythia timwn.
        b :IN std_logic_vector(N-1 downto 0);
-- Shma eijodoy poy lambanei mia akoloythia timwn.

        c :IN STD_LOGIC_VECTOR(N-1 downto 0);

        d :IN STD_LOGIC_VECTOR(N-1 downto 0);

        x :OUT std_logic_vector(N-1 downto 0));
END mux;

--Dhlwsh arxitektonikhs
ARCHITECTURE behavioral OF mux IS
BEGIN
  --Ylopoihsh toy pinaka alhtheias
  --dld analoga me thn timh toy sel
  --pairnei kai antistoixh timh toy pinaka to shma ejodoy
        WITH (sel) SELECT
                x <= a WHEN "00",
                         b WHEN "01",
                         c WHEN "10",
                         d WHEN "11",
                    (OTHERS=>'Z') WHEN others;

END behavioral;
```

---

bigLunit

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

ENTITY bigLunit IS
GENERIC(N: NATURAL :=120);
PORT(

 inp1 : IN std_logic_vector(N-1 DOWNTO 0);

 inp2 : IN std_logic_vector(N-1 DOWNTO 0);

 sel  : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
```

```
output : OUT std_logic_vector(N-1 DOWNTO 0));
END bigLunit;

ARCHITECTURE structural OF bigLunit IS

COMPONENT xor8 IS
        GENERIC(N: NATURAL := 8);
        PORT( A : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                  B : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                  Y : OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0));
                  END COMPONENT;

COMPONENT or16 IS
        GENERIC(N: NATURAL := 16);
        PORT( A : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                  B : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                  Y : OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0));
                  END COMPONENT;

COMPONENT nand32 IS
        GENERIC(N: NATURAL := 32);
        PORT( A : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                  B : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                  Y : OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0));
                  END COMPONENT;

COMPONENT nor64 IS
        GENERIC(N: NATURAL := 64);
        PORT( A : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                  B : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                  Y : OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0));
                  END COMPONENT;

COMPONENT mux IS
        GENERIC(N: NATURAL := 8);
        PORT(     a : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                  b : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                  c : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                  d : IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                  sel : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
                  x : OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0));
                  END COMPONENT;

                  SIGNAL outXor : STD_LOGIC_VECTOR(N-113 DOWNTO 0);
                  SIGNAL outOr : STD_LOGIC_VECTOR(N-97 DOWNTO 8);
```

```
                    SIGNAL outNand : STD_LOGIC_VECTOR(N-65 DOWNTO 24);
                    SIGNAL outNOr : STD_LOGIC_VECTOR(N-1 DOWNTO 56);

                    SIGNAl in0_mux : STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                    SIGNAL in1_mux : STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                    SIGNAL in2_mux : STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                    SIGNAL in3_mux : STD_LOGIC_VECTOR(N-1 DOWNTO 0);

BEGIN
 XOR_GATE:  xor8   GENERIC MAP(N-112)  PORT MAP(inp1(N-113 DOWNTO 0),inp2(N-113 DOWNTO 0),ou
 OR_GATE:   or16   GENERIC MAP(N-104)  PORT MAP(inp1(N-97 DOWNTO 8),inp2(N-97 DOWNTO 8),out(
 NAND_GATE: nand32 GENERIC MAP(N-88)   PORT MAP(inp1(N-65 DOWNTO 24),inp2(N-65 DOWNTO 24),ou
 NOR_GATE:  nor64  GENERIC MAP(N-56)   PORT MAP(inp1(N-1 DOWNTO 56),inp2(N-1 DOWNTO 56 ),out

        in0_mux <= (N-1 DOWNTO 8 =>'Z') & outXor;
        in1_mux <= (N-1 DOWNTO 24 =>'Z') & outOr & (7 DOWNTO 0 => 'Z');
        in2_mux <= (N-1 DOWNTO 56 =>'Z') & outNand & (23 DOWNTO 0 =>'Z');
        in3_mux <= outNOr & (55 DOWNTO 0 =>'Z');


 MUX_GATE: mux     GENERIC MAP(N)      PORT MAP(in0_mux,in1_mux,in2_mux,in3_mux,sel,output);
END structural;
```